



**THE UNITED STATES PATENT AND TRADEMARK OFFICE**

AF  
JPW

In re Patent Application of

BUTCHER et al

Atty. Ref.: 550-541; Confirmation No. 4617

Appl. No. 10/807,498

TC/A.U. 2183

Filed: March 24, 2004

Examiner: A. Li

For: NULL EXCEPTION HANDLING

\* \* \* \* \*

**Mail Stop Appeal Brief – Patents**

October 15, 2007

Commissioner for Patents

P.O. Box 1450

Alexandria, VA 22313-1450

Sir:

**RESPONSE TO NOTIFICATION OF NON-COMPLIANT APPEAL BRIEF**

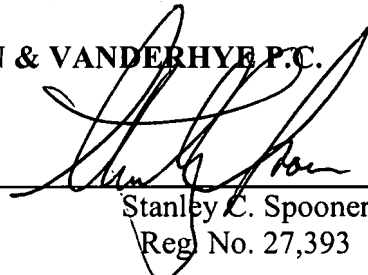
Responsive to the Notification of Non-Compliant Appeal Brief mailed September 18, 2007, submitted herewith is an amended Appeal Brief. The Appeal Brief has been revised to delete the claim identifiers from the Claim Appendix.

The newly submitted Appeal Brief is believed to overcome the issue identified in the Notification of Non-Compliant Appeal Brief. Appellants request that the appeal proceed on the merits.

Respectfully submitted,

**NIXON & VANDERHYE P.C.**

By: \_\_\_\_\_

  
Stanley C. Spooner  
Reg. No. 27,393

SCS:kmm  
901 North Glebe Road, 11th Floor  
Arlington, VA 22203-1808  
Telephone: (703) 816-4000  
Facsimile: (703) 816-4100



IN THE UNITED STATES PATENT AND TRADEMARK OFFICE  
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Patent Application of

Confirmation No.: 4617

BUTCHER et al

Atty. Ref.: 550-541

Serial No. 10/807,498

Group: 2183

Filed: March 24, 2004

Examiner: A. Li

For: NULL EXCEPTION HANDLING

\*\*\*\*\*

**APPEAL BRIEF**

On Appeal From Group Art Unit 2183

Stanley C. Spooner  
**NIXON & VANDERHYE P.C.**  
11<sup>th</sup> Floor, 901 North Glebe Road  
Arlington, Virginia 22203  
(703) 816-4028  
Attorney for Appellant



## TABLE OF CONTENTS

I. REAL PARTY IN INTEREST .....	1
II. RELATED APPEALS AND INTERFERENCES.....	1
III. STATUS OF CLAIMS .....	2
IV. STATUS OF AMENDMENTS.....	2
V. SUMMARY OF THE CLAIMED SUBJECT MATTER .....	2
VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL.....	7
VII. ARGUMENT .....	7
A. The Examiner fails to support his rejection of claims 31-60 under 35 USC §112 (first paragraph) as failing to comply with the enablement requirement .....	8
B. The Examiner fails to support his rejection of claims 31-60 under 35 USC §112 (second paragraph) as being indefinite .....	10
C. The Examiner fails to appreciate that the Click reference fails to teach claimed structures and method steps and claimed interrelationships .....	11
D. The Examiner fails to appreciate that the Smith reference fails to teach claimed structures and method steps and claimed interrelationships .....	12
E. The Examiner fails to provide any “reason” for combining the Click and Smith references .....	12
F. The Examiner apparently fails to appreciate that the Click and Smith references are mutually incompatible.....	13
G. The Examiner fails to appreciate that the Click reference would lead one of ordinary skill in the art away from the claimed invention.....	14
H. The Examiner fails to provide any evidentiary support for his rejection of claims under 35 USC §103 as being unpatentable over Click in view of Smith....	16

I. The Examiner fails to provide any evidentiary support for his rejection of claims under 35 USC §103 as being unpatentable over Click combined with Smith in view of Mirapuri ..... 17

VIII. CONCLUSION..... 18

IX. CLAIMS APPENDIX ..... A1

X. EVIDENCE APPENDIX..... A17

XI. RELATED PROCEEDINGS APPENDIX ..... A18



**THE UNITED STATES PATENT AND TRADEMARK OFFICE  
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES**

In re Patent Application of

BUTCHER et al

Atty. Ref.: 550-541

Serial No. 10/807,498

Group: 2183

Filed: March 24, 2004

Examiner: A. Li

For: NULL EXCEPTION HANDLING

\*\*\*\*\*

October 15, 2007

Mail Stop Appeal Brief - Patents  
Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

**APPEAL BRIEF**

Sir:

**I. REAL PARTY IN INTEREST**

The real party in interest in the above-identified appeal is ARM Limited by virtue of an assignment of rights from the inventors to ARM Limited recorded July 29, 2004 at Reel 15633, Frame 120.

**II. RELATED APPEALS AND INTERFERENCES**

There are believed to be no related appeals, interferences or judicial proceedings with respect to the present application, other than the Pre-Appeal Brief Request for Review previously filed in this appeal.

### **III. STATUS OF CLAIMS**

Claims 1-60 stand rejected in the Final Official Action and claims 61-63 have been cancelled. The Examiner contends that with respect to claims 31-60, they are rejectable as failing to comply with the enablement requirement of 35 USC §112 (first paragraph). Additionally, claims 31-60 are rejected under 35 USC §112 (second paragraph) as failing to particularly point out and distinctly claim the subject matter of the invention. Claims 1, 6-11, 14-16, 21-26, 29-31, 36-41, 44-46, 51-56 and 59-60 stand rejected under 35 USC §103 over Click (U.S. Patent 6,363,522) in view of Smith (U.S. Patent 5,430,862). Claims 2-5, 12, 13, 17-20, 27, 28, 32-35, 42, 43, 47-50, 57 and 58 stand rejected under 35 USC §103 as unpatentable over the Click/Smith combination, further in view of Mirapuri (U.S. Patent 5,590,294). The above rejections of claims 1-60 are appealed.

### **IV. STATUS OF AMENDMENTS**

No further response has been submitted with respect to the Final Official Action in this application other than the filing of a Pre-Appeal Brief Request for Review which decision was mailed on May 4, 2007.

### **V. SUMMARY OF THE CLAIMED SUBJECT MATTER**

Appellants' specification and figures provide an explanation of the claimed invention set out in independent claims 1, 16, 31 and 46, with each claimed

structure and method step addressed as to its location in the specification and in the figures.

“1. An apparatus for processing data [2 as shown in fig. 1 and discussed on page 7, lines 20-25 and elsewhere in the specification] comprising:

processing logic [register file 10, multiplier 12, shifter 14 and adder 16 as shown in fig. 1 and discussed on page 7, lines 20-25 and elsewhere in the specification] operable to perform data processing operations; and

an instruction decoder [18 as shown in fig. 1 and discussed on page 7, lines 27-33 and elsewhere in the specification] operable to decode program instructions to control said processing logic to perform data processing operations specified by said program instructions; wherein said instruction decoder, in response to a memory access instruction [discussed on page 8, lines 6-7 and elsewhere in the specification],

compares a base register value, stored within a base register specified by a base register field of said memory access instruction, with a predetermined null value [steps 26 and 28 as shown in fig. 3 and discussed on page 8, lines 31 to page 9, line 4 and elsewhere in the specification]; and,

if said base register value matches said predetermined null value, then said decoder triggers branching to execution of a null value exception handler [steps 32, 34 & 36 as shown in fig. 3 and discussed on page 9, lines 9-16 and elsewhere in the specification].”

“16. A method of processing data with an apparatus for processing data [2 as shown in fig. 1 and discussed on page 7, lines 20-25 and elsewhere in the specification] having processing logic [register file 10, multiplier 12, shifter 14 and adder 16 as shown in fig. 1 and discussed on page 7, lines 20-25 and elsewhere in the specification] operable to perform data processing operations and an instruction decoder [18 as shown in fig. 1 and discussed on page 7, lines 27-33 and elsewhere in the specification] operable to decode program instructions to control said processing logic to perform data processing operations specified by said program instructions, said method comprising the steps of:

in response to said memory access instruction decoded by said instruction decoder controlling said processing logic [discussed on page 8, lines 6-7 and elsewhere in the specification]

comparing a base register value [step 28], stored within a base register specified by a base register field of said memory access instruction [step 26], with a predetermined null value [steps 26 and 28 as shown in fig. 3 and discussed on page 8, lines 31 to page 9, line 4 and elsewhere in the specification]; and,

if said base register value matches said predetermined null value [step 36], then triggering branching to execution of a null value exception handler [steps 32, 34 & 36 as shown in fig. 3 and discussed on page 9, lines 9-16 and elsewhere in the specification].”



“31. A computer program product comprising a computer readable storage medium containing computer readable instructions for controlling an apparatus for processing data [2 as shown in fig. 1 and discussed on page 7, lines 20-25 and elsewhere in the specification] having processing logic [register file 10, multiplier 12, shifter 14 and adder 16 as shown in fig. 1 and discussed on page 7, lines 20-25 and elsewhere in the specification] operable to perform data processing operations and an instruction decoder [18 as shown in fig. 1 and discussed on page 7, lines 27-33 and elsewhere in the specification] operable to decode program instructions to control said processing logic to perform data processing operations specified by said program instructions, said computer program comprising,

in response to a memory access instruction decodable by said instruction decoder to control said processing logic [discussed on page 8, lines 6-7 and elsewhere in the specification],

comparing a base register value [step 28], stored within a base register specified by a base register field of said memory access instruction [step 26], with a predetermined null value [steps 26 and 28 as shown in fig. 3 and discussed on page 8, lines 31 to page 9, line 4 and elsewhere in the specification]; and,

if said base register value matches said predetermined null value [step 36], then branching to execution of a null value exception handler [steps 32, 34 & 36 as

shown in fig. 3 and discussed on page 9, lines 9-16 and elsewhere in the specification].”

“46. A computer program product comprising a computer readable storage medium containing computer readable instructions for translating non-native program instructions to form native program instructions directly decodable by an apparatus for processing data [2 as shown in fig. 1 and discussed on page 7, lines 20-25 and elsewhere in the specification] having processing logic [register file 10, multiplier 12, shifter 14 and adder 16 as shown in fig. 1 and discussed on page 7, lines 20-25 and elsewhere in the specification] operable to perform data processing operations and an instruction decoder [18 as shown in fig. 1 and discussed on page 7, lines 27-33 and elsewhere in the specification] operable to decode program instructions to control said processing logic to perform data processing operations specified by said program instructions, said native program instructions comprising,

in response to a memory access instruction decodable by said instruction decoder to control said processing logic [discussed on page 8, lines 6-7 and elsewhere in the specification]

comparing a base register value [28], stored within a base register specified by a base register field of said memory access instruction [26], with a

predetermined null value [steps 26 and 28 as shown in fig. 3 and discussed on page 8, lines 31 to page 9, line 4 and elsewhere in the specification]; and,

if said base register value matches said predetermined null value [step 36], then triggering branching to execution of a null value exception handler [steps 32, 34 & 36 as shown in fig. 3 and discussed on page 9, lines 9-16 and elsewhere in the specification].”

## **VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL**

Claims 31-60 stand rejected under 35 USC §112 (first paragraph) as failing to comply with the enablement requirement.

Claims 31-60 stand rejected under 35 USC §112 (second paragraph) as failing to particularly point out and distinctly claim the subject matter of the invention.

Claims 1, 6-11, 14-16, 21-26, 29-31, 36-41, 44-46, 51-56, 59 and 60 stand rejected under 35 USC §103 as unpatentable over Click in view of Smith.

Claims 2-5, 12, 13, 17-20, 27, 28, 32-35, 42, 43, 47-50, 57 and 58 stand rejected under 35 USC §103 as unpatentable over Click, Smith and Mirapuri.

## **VII. ARGUMENT**

Appellants’ arguments include the fact that the burden is on the Examiner to first and foremost properly construe the language of the claims to determine

what structure and/or method steps are covered by that claim. After proper construction of the claim language, the burden is also on the Examiner to demonstrate where a single reference (in the case of anticipation) or a plurality of references (in the case of an obviousness rejection) teaches each of the structures and/or method steps recited in independent claims 1, 16, 31 and 46.

Furthermore, the Court of Appeals for the Federal Circuit has stated in the case of *In re Rouffet*, 47 USPQ2d 1453, 1458 (Fed. Cir. 1998)

to prevent the use of hindsight based on the invention to defeat patentability of the invention, this court **requires** the examiner to show a **motivation** to combine the references that create the case of obviousness. In other words, the Examiner **must show reasons** that the skilled artisan, confronted with the same problems as the inventor and with no knowledge of the claimed invention, would select the elements from the cited prior art references for combination in the manner claimed. (Emphasis added).

**A. The Examiner fails to support his rejection of claims 31-60 under 35 USC §112 (first paragraph) as failing to comply with the enablement requirement**

The Examiner on page 2 of the Final Rejection alleges that, with respect to claims 31-60, the specification fails to provide an enabling disclosure under 35 USC §112(1<sup>st</sup> ¶). In detailing his alleged basis, the Examiner states that “[t]he specification does not describe how it is possible for a computer program product,

such as a series of instructions forming a computer program, is [sic] able to contain a computer readable storage medium” (emphasis added). This statement misstates independent claims 31 and 46, and claims 32-45 and 47-60, respectively, dependent thereon.

The actual language of the independent claims is “A computer program product comprising a computer readable storage medium containing computer readable instructions . . . .” (emphasis added, independent claims 31 and 46). The Examiner misquotes the subject independent claims as requiring a Computer program product . . . to contain a computer readable storage medium” whereas the claim requires the computer readable storage medium to contain “computer readable instructions.”

Because the Examiner misquotes independent claims 31 and 46, he misunderstands what claimed structure is required to be enabled by Appellants specification and therefore fails to provide any basis for rejection of claims 31-60 under 35 USC §112(1<sup>st</sup> ¶). There is no allegation that Appellants specification does not provide an enabling disclosure of the claimed “computer readable storage medium containing computer readable instructions.”

**B. The Examiner fails to support his rejection of claims 31-60 under 35 USC §112 (second paragraph) as being indefinite**

The Examiner rejects claims 31-60 under 35 USC §112(2<sup>nd</sup> ¶) as being indefinite, stating that it is “unclear how a computer program product contains a computer readable storage medium” (emphasis added). The actual claim language in the independent computer program products claims 31 and 46 states “a computer program product comprising a computer readable storage medium . . . .” Thus, Appellants’ claims accurately state that the computer program product includes, but is not limited to, a “computer readable storage medium” which limitations meet the Patent Office imposed requirement for a “tangible result” with respect to computer program product claims.

As with the rejection of claims 31-60 under 35 USC §112(1<sup>st</sup> ¶), Examiner misstates Appellants’ claim language. The correct language of the independent claims was previously noted in the last Amendment (page 18), and such language has routinely been accepted by the Patent Office (see U.S. Patent 6,836,860) and therefore does not evidence any indefiniteness.

**C. The Examiner fails to appreciate that the Click reference fails to teach claimed structures and method steps and claimed interrelationships**

In the outstanding Final Rejection, the Examiner admits that “Click has not explicitly taught . . . a. Processing logic . . . and b. An instruction decoder . . .” as recited in independent claims 1, 16, 31 and 46 (See section 11, subsections a. and b. on page 4 of the Final Rejection). These admissions are very much appreciated and are dispositive confirmation that these elements are missing from the Click reference.

The Examiner then suggests that the interrelationship between these missing elements is somehow disclosed in the Click reference (“However, Click has taught that the memory access instructions are executed on a processor, as shown in Figure 5, but provides no details about the ‘N Processor’.” Simple logic dictates that, if the structures are not taught in Click (as admitted by the Examiner), the claimed interrelationship between those unclaimed structures cannot possibly be taught.

In view of the Examiner’s admissions, Click cannot possibly teach the claimed interrelationship between the “processing logic” and the “instruction decoder” structures given that the structures aren’t present in Click.

**D. The Examiner fails to appreciate that the Smith reference fails to teach claimed structures and method steps and claimed interrelationships**

In citing the Smith reference the Examiner make only a general allegation that Smith teaches the “processing logic” and the “instruction decoder” which he admits is missing from Click reference. However, the Examiner disregards the details of the “instruction decoder” which are specified in the last portion of the independent claims, i.e., from the “wherein said instruction decoder, in response to said memory access instruction, compares . . . ; and, if . . . , then said decoder triggers branching to execution of a null value exception handler.” (claim 1 with similar language in claims 16, 31 and 46).

Because the Examiner does not allege that the above claim language is disclosed in Smith, this is taken as an admission that it is missing from Smith as well as Click.

**E. The Examiner fails to provide any “reason” for combining the Click and Smith references**

As consistently stated by the Court of Appeals for the Federal Circuit and as confirmed in a Memorandum from Deputy Commissioner for Patent Operations, Margaret A. Focarino on May 3, 2007, “in formulating a rejection under 35 USC §103(a) based upon a combination of prior art elements, it remains



necessary [for the examiner] to identify the reasons why a person of ordinary skill in the art would have combined the prior art elements in the manner claimed.”

The PTO position is confirmed by the Federal Circuit decision in the *In re Rouffet* case noted above, and it is noted that at no point in the Official Action does the Examiner provide any “reason” or “motivation” for combining bits and pieces of the cited prior art references.

While the Examiner does suggest that increased process performance and compatibility would be a motivation, this is a mere conclusory statement only. The Examiner has provided no evidence of record establishing any “reason” or “motivation.”

Therefore, the failure to provide any reason for combining the Click and Smith references is a failure in any obviousness rejection based thereon.

**F. The Examiner apparently fails to appreciate that the Click and Smith references are mutually incompatible**

The Examiner alleges that it would be obvious to “incorporate the processor of Smith in the device of Click to increase processor performance and capability.” However, if one of ordinary skill in the art followed the teachings of Smith and did not use a translating program, then he would not be able to identify the redundant null checks in advance and remove them as is taught (and required) by the Click reference.

Click relies upon an analysis made during compilation to remove redundant null checks from the code as it is being compiled from source code form into runtime bytecodes (as clearly disclosed in Figure 6 of Click). In Smith, at column 1, lines 16-20, he discusses the use of separate translating programs (such as compilers) to convert source computer programs to an executable form. Smith then dismisses this approach as too time consuming and inefficient.

Also, using a separate translating program such as disclosed in Click removes the need for any hardware-based conversion, such as in Smith. The Click and Smith references are simply mutually incompatible and this incompatibility would be clearly obvious to those of ordinary skill in the art.

The Examiner's assertion that one of ordinary skill in the art would be motivated to combine Click and Smith for the stated generic benefits of "increase[d] processor performance and capability" fails to address the basic reality that the teaching of these two documents is fundamentally incompatible and contradictory. The Click and Smith references are mutually incompatible and therefore cannot and would not be combined as suggested by the Examiner.

**G. The Examiner fails to appreciate that the Click reference would lead one of ordinary skill in the art away from the claimed invention**

Because the Examiner admits that it fails to teach the structure and yet the Examiner contends it teaches the structural interrelationship, Click would appear

to lead one of ordinary skill in the art away from the structures recited in Appellants' claims.

The Examiner's citation of Tanenbaum (Final Rejection, page 14) is unclear as it is not combined in any rejection under 35 USC §103. If one assumes that it is to be combined with Click and Smith combination, one would clearly recognize that the decision regarding whether functionality is to be executed in hardware or software has a profound influence upon real-life performance. Tanenbaum specifically states at page 11, paragraph 4, "the decision to put certain functions in hardware and others in software is based upon such factors as cost, speed, reliability, and frequency of expected changes."

The Click reference, as the Examiner contends, teaches that less null checks means smaller and faster code so that one should remove null checks. Appellants admit that this was common knowledge that pre-dates even the Click reference. However, Appellants' claimed invention does not specifically remove null checks and, in fact, because of the way the claimed invention works, Appellants invention actually add more. Every load and store in the instruction sets performs a null check.

Given the Click teaching, the claimed invention should not actually provide an advantage and this would be clear to those of ordinary skill reviewing the Click reference. Yet, when implemented in hardware in accordance with the present invention, it does provide an advantage. Click and its teachings would clearly lead

one of ordinary skill in the art away from utilizing hardware for null checks as set out in Appellants' claims.

**H. The Examiner fails to provide any evidentiary support for his rejection of claims under 35 USC §103 as being unpatentable over Click in view of Smith**

Because the traversal of both obviousness rejections rely upon numerous previously discussed sections, those sections will be incorporated by reference.

As stated in numerous Federal Circuit case decisions, “the PTO has the burden under Section 103 to establish a *prima facie* case of obviousness.” *In re Fine*, 5 USPQ2d 1596, 1598 (Fed. Cir. 1988). “It can satisfy this burden only by showing some objective teaching in the prior art or that knowledge generally available to one of ordinary skill in the art would lead that individual to combine the relevant teachings of the references” (emphasis added). As noted above in section C, the Examiner admits that Click fails to teach the claimed “instruction decoder” and the claimed interrelationships between it and other claimed elements. As noted in section D, the Examiner fails to indicate where the Smith reference teaches the claimed “instruction decoder” and especially its claimed interrelationship. Therefore, even if Click and Smith were combined, they would fail to teach all claimed elements and interrelationships and thus the rejections of the independent claims under 35 USC §103 fails.

Further, as discussed in section E, the Examiner fails to meet his burden of providing, identifying or articulating any “reason” or “motivation for combining

the Click and Smith references and so the §103 rejection fails. Additionally, as noted in section F above, the Click and Smith teachings, as would be known by those of ordinary skill in the art, are incompatible and could not be combined. Finally, as discussed in section G, the Click reference clearly would lead one of ordinary skill away from the claimed invention. As the Federal Circuit has decided, it is “error to find obviousness where references ‘diverge from and teach away from the invention at hand’.” *Id.*

Thus, for all of the above reasons, the Examiner has failed to present any prima facie case of obviousness under 35 USC §103 of independent claims 1, 16, 31 and 46 or claims dependent thereon over the Click and Smith combination of references.

**I. The Examiner fails to provide any evidentiary support for his rejection of claims under 35 USC §103 as being unpatentable over Click combined with Smith in view of Mirapuri**

The arguments set out in section H above as well as the sections cited therein are herein incorporated by reference with respect to the combination of the Click and Smith references.

The Examiner fails to indicate that the Mirapuri reference teaches the “instruction decoder” which is missing from the Click and Smith references and therefore none of the references teach this claimed structure and claimed

interrelationship. The Examiner fails to indicate how or why Mirapuri contains any “reason” or “motivation” for combining the Click, Smith and Mirapuri references in the manner of any of the pending claims.

For these and the previously noted reasons in the above sections, the Examiner has failed to present any *prima facie* case of obviousness under 35 USC §103 of independent claims 1, 16, 31 and 46 or claims dependent thereon over the Click, Smith and Mirapuri combination of references

### **VIII. CONCLUSION**

As discussed in detail above, no cited prior art reference teaches the claimed “instruction decoder” (as admitted by the Examiner in regard to Click and by the failure to identify any corresponding disclosure in Smith or Mirapuri). No reference or other document contains any “reason” or “motivation” to combine two or more of the cited references in the manner of the independent claims. In fact, it is demonstrated not only that the Click and Smith references are incompatible, it has also been shown that the Click reference teaches away from the Click/Smith combination. For numerous reasons, the Examiner has simply failed to set out a *prima facie* basis for rejection.

As a result of the above, there is simply no support for the rejections of Applicants' independent claims or claims dependent thereon under 35 USC §112 or §103. Thus, and in view of the above, the rejection of claims 1-60 under 35

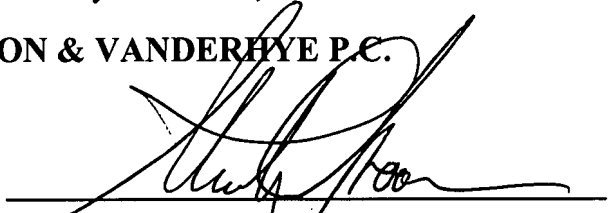
BUTCHER et al  
Serial No. 10/807,498

USC §§112 and 103 is clearly in error and reversal thereof by this Honorable  
Board is respectfully requested.

Respectfully submitted,

**NIXON & VANDERHYTE P.C.**

By:

  
\_\_\_\_\_  
Stanley C. Spooner  
Reg. No. 27,393

SCS:kmm  
Enclosure



## IX. CLAIMS APPENDIX

1. An apparatus for processing data comprising:  
  
processing logic operable to perform data processing operations; and  
  
an instruction decoder operable to decode program instructions to control  
  
said processing logic to perform data processing operations specified by said  
  
program instructions; wherein said instruction decoder, in response to a memory  
  
access instruction, compares a base register value, stored within a base register  
  
specified by a base register field of said memory access instruction, with a  
  
predetermined null value; and, if said base register value matches said  
  
predetermined null value, then said decoder triggers branching to execution of a  
  
null value exception handler.
  
2. An apparatus as claimed in claim 1, wherein in response to said memory  
  
access instruction a return address is stored pointing a memory location storing a  
  
program instruction to be executed upon a return from said null value exception  
  
handler.
  
3. An apparatus as claimed in claim 1, wherein said null value exception  
  
handler is located at a memory address pointed to by a value stored within a  
  
programmable configuration register.



4. An apparatus as claimed in claim 3, wherein said programmable configuration register is a coprocessor configuration register.

5. An apparatus as claimed in claim 3, wherein said branch is made to an instruction stored at a memory address given by said value stored within said programmable configuration register subject to a fixed offset.

6. An apparatus as claimed in claim 1, wherein said null value exception handler is operable to determine if said memory access instruction attempting to access a location corresponding to a null value corresponds to emulation of a non-native program instruction that is not directly decodable by said instruction decoder attempting to make a memory access using a null value.

7. An apparatus as claimed in claim 1, wherein said null value exception handler is operable to determine if said memory access instruction attempting to access a location corresponding to a null value corresponds to an error in operation of a virtual machine computer program operable to translate non-native program instructions that are not directly decodable by said instruction decoder into native program instructions that are directly decodable by said instruction decoder.

8. An apparatus as claimed in claim 6, wherein said non-native program instructions are machine independent program instructions.

9. An apparatus as claimed in claim 8, wherein said machine independent program instructions are one of:

Java bytecodes;

MSIL bytecodes;

CIL bytecodes; and

.NET bytecodes.

10. An apparatus as claimed in claim 6, wherein said non-native instructions are native program instructions of a different apparatus for processing data.

11. An apparatus as claimed in claim 10, wherein said processing logic and said instruction decoder are part of a RISC processor and said non-native instructions are native instructions of a CISC processor.

12. An apparatus as claimed in claim 3, wherein said value stored within said programmable configuration register is a start address of said null value exception handler.

13. An apparatus as claimed in claim 3, wherein said value stored within said programmable configuration register is an address of a jump instruction operable to jump execution to a start address of said null value exception handler.

14. An apparatus as claimed in claim 1, wherein said memory access instruction is a load instruction operable to load into a destination register specified by a destination register field within said load instruction a load value dependent upon a value read from a memory location specified by said base register value.

15. An apparatus as claimed in claim 1, wherein said memory access instruction is a store instruction operable to store into a memory location specified by said base register value a store value dependent upon source value stored within a source register specified by a source register field within said store instruction.

16. A method of processing data with an apparatus for processing data having processing logic operable to perform data processing operations and an instruction decoder operable to decode program instructions to control said

processing logic to perform data processing operations specified by said program instructions, said method comprising the steps of:

in response to said memory access instruction decoded by said instruction decoder controlling said processing logic comparing a base register value, stored within a base register specified by a base register field of said memory access instruction, with a predetermined null value; and, if said base register value matches said predetermined null value, then triggering branching to execution of a null value exception handler.

17. A method as claimed in claim 16, wherein in response to said memory access instruction a return address is stored pointing a memory location storing a program instruction to be executed upon a return from said null value exception handler.

18. A method as claimed in claim 16, wherein said null value exception handler is located at a memory address pointed to by a value stored within a programmable configuration register.

19. A method as claimed in claim 18, wherein said programmable configuration register is a coprocessor configuration register.

20. A method as claimed in claim 18, wherein said branch is made to an instruction stored at a memory address given by said value stored within said programmable configuration register subject to a fixed offset.

21. A method as claimed in claim 16, wherein said null value exception handler is operable to determine if said memory access instruction attempting to access a location corresponding to a null value corresponds to emulation of a non-native program instruction that is not directly decodable by said instruction decoder attempting to make a memory access using a null value.

22. A method as claimed in claim 16, wherein said null value exception handler is operable to determine if said memory access instruction attempting to access a location corresponding to a null value corresponds to an error in operation of a virtual machine computer program operable to translate non-native program instructions that are not directly decodable by said instruction decoder into native program instructions that are directly decodable by said instruction decoder.

23. A method as claimed in claim 21, wherein said non-native program instructions are machine independent program instructions.

24. A method as claimed in claim 23, wherein said machine independent program instructions are one of:

Java bytecodes;  
MSIL bytecodes;  
CIL bytecodes; and  
.NET bytecodes.

25. A method as claimed in claim 21, wherein said non-native instructions are native program instructions of a different apparatus for processing data.

26. A method as claimed in claim 25, wherein said processing logic and said instruction decoder are part of a RISC processor and said non-native instructions are native instructions of a CISC processor.

27. A method as claimed in claim 18, wherein said value stored within said programmable configuration register is a start address of said null value exception handler.

28. A method as claimed in claim 18, wherein said value stored within said programmable configuration register is an address of a jump instruction operable to jump execution to a start address of said null value exception handler.

29. A method as claimed in claim 16, wherein said memory access instruction is a load instruction operable to load into a destination register specified by a destination register field within said load instruction a load value dependent upon a value read from a memory location specified by said base register value.

30. A method as claimed in claim 16, wherein said memory access instruction is a store instruction operable to store into a memory location specified by said base register value a store value dependent upon source value stored within a source register specified by a source register field within said store instruction.

31. A computer program product comprising a computer readable storage medium containing computer readable instructions for controlling an apparatus for processing data having processing logic operable to perform data processing operations and an instruction decoder operable to decode program instructions to control said processing logic to perform data processing operations specified by said program instructions, said computer program comprising, in response to a memory access instruction decodable by said instruction decoder to control said processing logic, comparing a base register value, stored within a base register

specified by a base register field of said memory access instruction, with a predetermined null value; and, if said base register value matches said predetermined null value, then branching to execution of a null value exception handler.

32. A computer program product as claimed in claim 31, wherein in response to said memory access instruction a return address is stored pointing a memory location storing a program instruction to be executed upon a return from said null value exception handler.

33. A computer program product as claimed in claim 31, wherein said null value exception handler is located at a memory address pointed to by a value stored within a programmable configuration register.

34. A computer program product as claimed in claim 33, wherein said programmable configuration register is a coprocessor configuration register.

35. A computer program product as claimed in claim 33, wherein said branch is made to an instruction stored at a memory address given by said value stored within said programmable configuration register subject to a fixed offset.



36. A computer program product as claimed in claim 31, wherein said null value exception handler is operable to determine if said memory access instruction attempting to access a location corresponding to a null value corresponds to emulation of a non-native program instruction that is not directly decodable by said instruction decoder attempting to make a memory access using a null value.

37. A computer program product as claimed in claim 31, wherein said null value exception handler is operable to determine if said memory access instruction attempting to access a location corresponding to a null value corresponds to an error in operation of a virtual machine computer program operable to translate non-native program instructions that are not directly decodable by said instruction decoder into native program instructions that are directly decodable by said instruction decoder.

38. A computer program product as claimed in claim 36, wherein said non-native program instructions are machine independent program instructions.

39. A computer program product as claimed in claim 38, wherein said machine independent program instructions are one of:

Java bytecodes;

MSIL bytecodes;

CIL bytecodes; and  
.NET bytecodes.

40. A computer program product as claimed in claim 36, wherein said non-native instructions are native program instructions of a different apparatus for processing data.

41. A computer program product as claimed in claim 40, wherein said processing logic and said instruction decoder are part of a RISC processor and said non-native instructions are native instructions of a CISC processor.

42. A computer program product as claimed in claim 33, wherein said value stored within said programmable configuration register is a start address of said null value exception handler.

43. A computer program product as claimed in claim 33, wherein said value stored within said programmable configuration register is an address of a jump instruction operable to jump execution to a start address of said null value exception handler.

44. A computer program product as claimed in claim 31, wherein said memory access instruction is a load instruction operable to load into a destination register specified by a destination register field within said load instruction a load value dependent upon a value read from a memory location specified by said base register value.

45. A computer program product as claimed in claim 31, wherein said memory access instruction is a store instruction operable to store into a memory location specified by said base register value a store value dependent upon source value stored within a source register specified by a source register field within said store instruction.

46. A computer program product comprising a computer readable storage medium containing computer readable instructions for translating non-native program instructions to form native program instructions directly decodable by an apparatus for processing data having processing logic operable to perform data processing operations and an instruction decoder operable to decode program instructions to control said processing logic to perform data processing operations specified by said program instructions, said native program instructions comprising, in response to a memory access instruction decodable by said instruction decoder to control said processing logic comparing a base register

value, stored within a base register specified by a base register field of said memory access instruction, with a predetermined null value; and, if said base register value matches said predetermined null value, then triggering branching to execution of a null value exception handler.

47. A computer program product as claimed in claim 46, wherein in response to said memory access instruction a return address is stored pointing a memory location storing a program instruction to be executed upon a return from said null value exception handler.

48. A computer program product as claimed in claim 46, wherein said null value exception handler is located at a memory address pointed to by a value stored within a programmable configuration register.

49. A computer program product as claimed in claim 48, wherein said programmable configuration register is a coprocessor configuration register.

50. A computer program product as claimed in claim 48, wherein said branch is made to an instruction stored at a memory address given by said value stored within said programmable configuration register subject to a fixed offset.

51. A computer program product as claimed in claim 46, wherein said null value exception handler is operable to determine if said memory access instruction attempting to access a location corresponding to a null value corresponds to emulation of a non-native program instruction that is not directly decodable by said instruction decoder attempting to make a memory access using a null value.

52. A computer program product as claimed in claim 46, wherein said null value exception handler is operable to determine if said memory access instruction attempting to access a location corresponding to a null value corresponds to an error in operation of a virtual machine computer program operable to translate non-native program instructions that are not directly decodable by said instruction decoder into native program instructions that are directly decodable by said instruction decoder.

53. A computer program product as claimed in claim 51, wherein said non-native program instructions are machine independent program instructions.

54. A computer program product as claimed in claim 53, wherein said machine independent program instructions are one of:

Java bytecodes;  
MSIL bytecodes;

CIL bytecodes; and  
.NET bytecodes.

55. A computer program product as claimed in claim 51, wherein said non-native instructions are native program instructions of a different apparatus for processing data.

56. A computer program product as claimed in claim 55, wherein said processing logic and said instruction decoder are part of a RISC processor and said non-native instructions are native instructions of a CISC processor.

57. A computer program product as claimed in claim 48, wherein said value stored within said programmable configuration register is a start address of said null value exception handler.

58. A computer program product as claimed in claim 48, wherein said value stored within said programmable configuration register is an address of a jump instruction operable to jump execution to a start address of said null value exception handler.

59. A computer program product as claimed in claim 46, wherein said memory access instruction is a load instruction operable to load into a destination register specified by a destination register field within said load instruction a load value dependent upon a value read from a memory location specified by said base register value.

60. A computer program product as claimed in claim 46, wherein said memory access instruction is a store instruction operable to store into a memory location specified by said base register value a store value dependent upon source value stored within a source register specified by a source register field within said store instruction.

**X. EVIDENCE APPENDIX**

None.



**XI. RELATED PROCEEDINGS APPENDIX**

None.